网络空间安全创新创业实践备注

小组成员：邵帅鸣 学号：202000460087

Github账号：beimuchengge

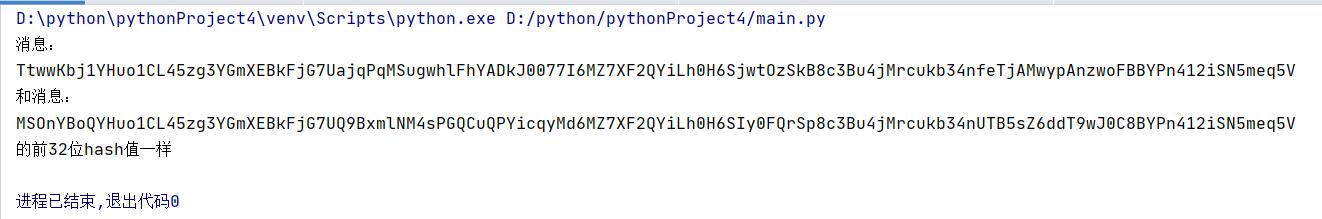
完成项目

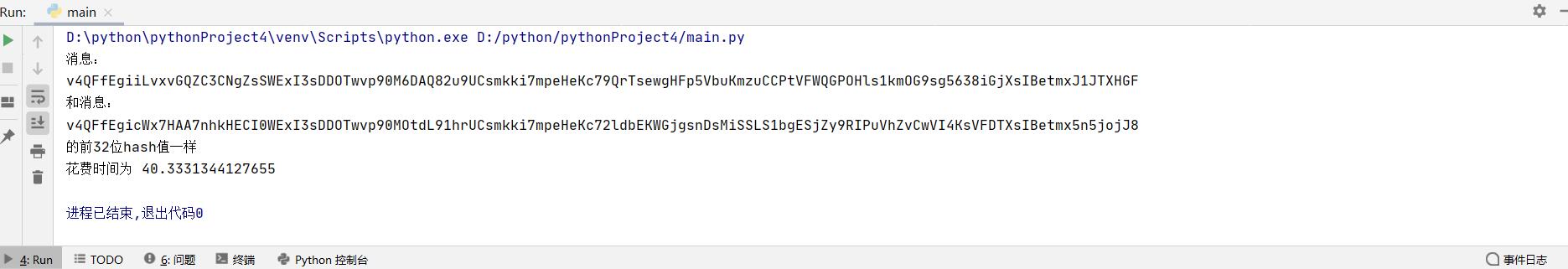
1. Sm3生日攻击

完成人：邵帅鸣

代码文件：sm3birthdayattack.txt（初版），sm3birthdayattacktime.txt（加入时间测试）

运行结果：sm3birthdayattack.jpg（初版）sm3birthdayattacktimejpg（加入时间结果）





运行环境：python

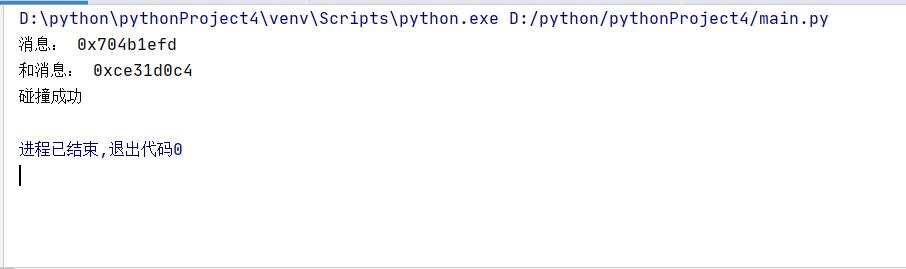
Ps：使用的库为pysmx.SM3 ，由于python在算法的优化不够完善，运行100次该算法消耗时间约为0.03s，平均运行一次时间为0.0003s，同比在最终优化后的c++sm3算法，平均运行1000w次时间约为45s，平均运行一次时间为0.0000045s，加速了约100倍，而在python环境下，我们进行32位的生日攻击运算可以在一分钟内得出结果，而如果使用优化后的sm3，则可达到秒出结果的效果。

1. Sm3rho算法

完成人：邵帅鸣

代码文件：sm3rho.txt（初版），sm3rhotime.txt(加入时间测试)

运行结果：sm3rho.jpg，sm3rhotime.jpg（加入时间测试结果）



运行环境：python

实验流程：生成一个32bit的随机数，进行sm3加密。然后取前32bit作为下一次加密的输入，以此类推。

然后按照rho算法的思想，判断其中是否有环出现，如果找到i使得hash[i]==hash[2\*i]，那么就向前寻找j使得hash[i-j+1]==hash[2\*i-j+1],但是hash[i-j]!=hash[2\*i-j]，则碰撞成功。

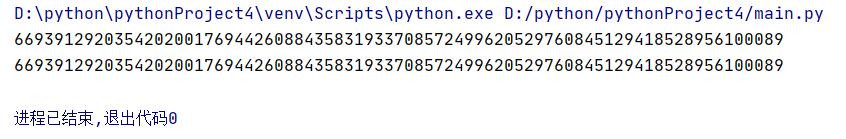
Ps：由于库中sm3的输入只能为字符串，故本次实验使用的sm3算法为本人自行在python上编写，代码文件为sm3

1. Sm3长度扩展攻击

完成人：邵帅鸣

代码文件：sm3lengthextensionattack.txt

运行结果：sm3lengthextensionattack.jpg



运行环境：python

实验流程：

plaintext是未知的，但是我们已知plaintext填充后进行sm3hash后的16进制数，m是我们想要置于padding后面的信息，这里我们选择把sm3enc（plaintext）作为第二次的IV值加密m，即可得到sm3enc（plaintext+padding+m）的值。

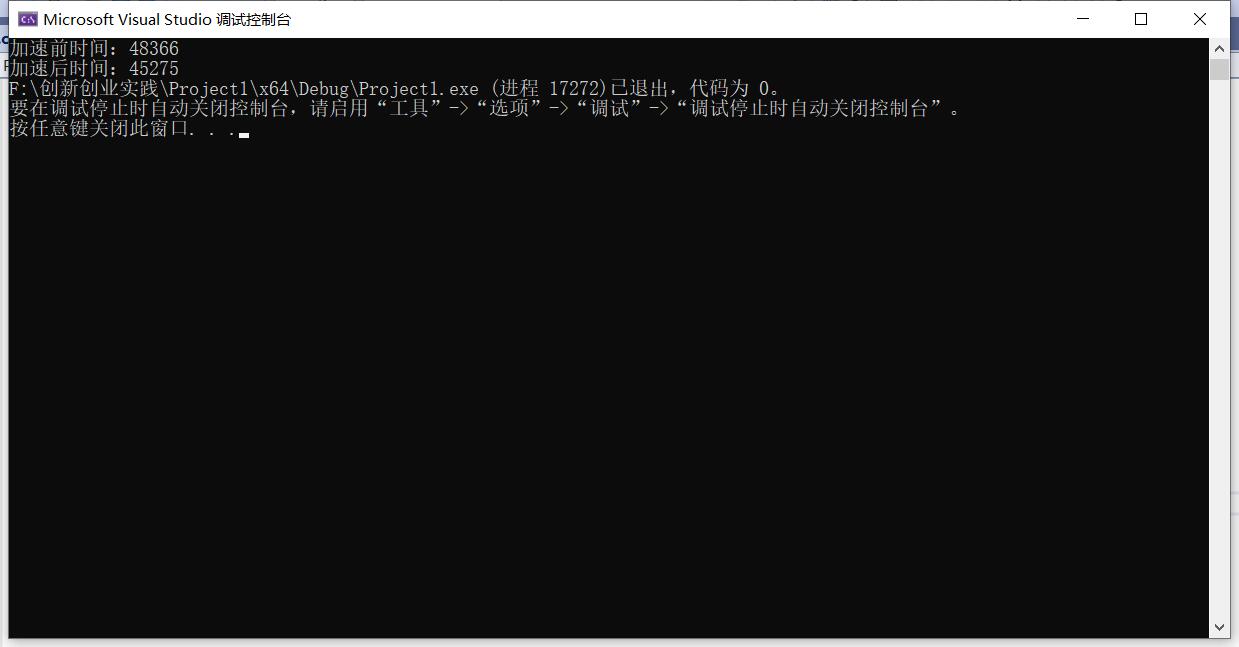
ps：由于在长度扩展攻击中，需要更改IV的值，所以我在本次的实验中对上一个实验写的sm3函数进行了更改，加入了IV的传参，在后续的实验中，我发现其实可以不需要加入IV进行传参，只需要先异或IV的默认值，在异或需要设置的IV值即可

1. Sm3算法加速优化

完成人：邵帅鸣

代码文件：c++sm3.txt

运行结果：c++sm3jpg



运行环境：c++

实验思路：在本次实验中，我实现了以下优化：

（1）对于W[]的计算过程，我发现可以将循环展开为步长为3的循环，每次同时对三个W进行计算，而且三个W的计算互不依赖，达到第一步的加速

（2）同时在同时计算三个W的计算过程中，我们使用SIMD，将原本需要进行三次的位运算在一次SIMD运算中解决，达到了第二步的加速。